This project has received funding from the EU Erasmus+ Programme under grant agreement 2021-1-EL02-KA220-YOU-000028780

European Commission

Co-funded by the Erasmus+ Programme of the European Union

BRIDGING THE GAP

An AI-enabled versatile skill matching tool to assist the less privileged

BRIDGING THE GAP DELIVERABLE 2.2

Authors:	MY COMPANY PROJECTS O.E., International Hellenic University
Status:	Final
Due Date:	31/12/2023
Version:	1.0
Dissemination Level:	PU

Disclaimer:

The content of this document was issued within the frame of the BRIDGING THE GAP project and represents the views of the authors only and is his/her sole responsibility. The European Union/ National agency does not accept any responsibility for use that may be made of the information it contains. The project has received funding from the European Union's Erasmus+: KA220-YOU under Grant Agreement 2021-1-EL02-KA220-YOU-000028780. This document and its content are the property of the BRIDGING THE GAP Consortium. All rights relevant to this document are determined by the applicable laws. Access to this document does not grant any right or license on the document or its contents. This document or its contents are not to be used or treated in any manner inconsistent with the rights or interests of the BRIDGING THE GAP Consortium or the Partners detriment and are not to be disclosed externally without prior written consent from the BRIDGING THE GAP Partners. Each BRIDGING THE GAP Partner may use this document in conformity with the BRIDGING THE GAP Consortium Grant Agreement provisions.

(*) Dissemination level. -PU: Public, fully open, e.g. web; CO: Confidential, restricted under conditions set out in Model Grant Agreement; CI: Classified, Int = Internal Working Document, information as referred to in Commission Decision 2001/844/EC.

BRIDGING THE GAP Project Profile

Grant Agreement No.:	2021-1-EL02-KA220-YOU-000028780
Acronym:	BRIDGING THE GAP
Title:	Bridging the Gap – An AI-enabled versatile skill matching tool to assist the less privileged
URL:	https://bridgingthegapproject.eu/
Start Date:	14/02/2022
Duration:	24 months

Partners

INTERNATIONAL HELLENIC UNIVERSITY	DIETHNES PANEPISTIMIO ELLADOS (IHU)	Greece
MYCOMPANY	MY COMPANY PROJECTS O.E.	Greece
	UNIVERSITATEA DIN CRAIOVA	Romania
	Regional center for vocational training and education to CCI-Blagoevgrad	Bulgaria

Document History

Version	Date	Author (Partner)	Remarks/Changes
0.1	04/12/2023	Kalliopi Kravari (IHU)	Table of Contents
0.2	11/12/2023	Dimitrios Sarafis (My Company Projects O.E.)	1st Draft ready for internal review
0.3	28/12/2023	Kalliopi Kravari (IHU)	2nd Draft ready for quality control
1.0	29/12/2023	Periklis Chatzimisios (IHU)	FINAL VERSION TO BE SUBMITTED

Abbreviations and acronyms

Deliverable	D
Expected Outcomes	EO
International Hellenic University	IHU
Non-governmental organization	NGO
Labour Force Survey	LFS
Neither in employment nor in education or training	NEET
Human Resources	HR
URI	Uniform Resource Identifier
CV	Curriculum Vitae
JV	Job Vacancy
UoL	Unit of Learning

Executive Summary

BRIDGING THE GAP is a 24 month duration project funding from the European Union's Erasmus+: KA220-YOU under Grant Agreement 2021-1-EL02-KA220-YOU-000028780.

The overarching objective of the BRIDGING THE GAP project is to provide a holistic approach beyond a classical skill-matching system to a system that will bridge the gap as to who are the underprivileged and why are they underprivileged and how education and skill improvement will benefit them.

The main purpose of this document is to a report the progress of the BRIDGING THE GAP project during the deliverable 2.2. More specifically, this deliverable reports on Project Results 2 findings that study a solution framework for the EU Company to annotate available jobs. The current document analyses the development of a solution framework for the EU Company to annotate available jobs that will help companies to annotate their job offers accessing more suitable candidates based on the skills provided in their CV.

Table of Contents

List of Figures & Tables

1 Introduction

1.1 Purpose of the document

The purpose of this document is to present the progress of the project during the second implementation phase regarding the implemented research activities as they are reported in Project Results 2 (deliverable 2.2).

1.2 Intended audience

The intended audience of this document consists of the following target groups:

- BRIDGING THE GAP project partners and the Project Officer at the National Agency
- Young people, especially on the Balkan area, that are interested in skill building/matching
- Labour market actors
- Universities, course providers

1.3 Work Package Objective

The current deliverable 2.2, part of Project Results 2, analyses the development of a solution framework for the EU Company to annotate available jobs, enabling them to find candidates more suitable for their needs by improving the job vacancies descriptions and by incorporating them in our common framework. More specifically, Project Results 2 are related to tools development and integrations with external systems & semantic intelligent bridging. For this purpose, all related deliverables including the current 2.2 focus on stakeholders, use cases serving them and building simple user interfaces through prototypes. These prototypes serve as proof of concept that CVs, job descriptions and courses can be annotated with semantics from standard specifications like ESCO. Moreover, we focused on technical issues and developed the back-end of our solution, an intelligent module that matches CVs and job, CVs and courses, Jobs and courses. The rules behind it are coded into this module. Towards this direction, we provide to the public a glass box interface explaining the way the match occurred. Moreover, we incorporate an intelligent software component in order to better guide and support at EU level the validation of non-formal and informal learning should be further pursued especially with regards to the validation of learning acquired through digital resources.

1.4 Structure of the document

In chapter 2, this report describes a short summary of the problem statement and the framework overview as defined by the Bridging The Gap project.

In chapter 3, this report discusses issues related to all the necessary data required by the developed framework, i.e. JVs scraped from multiple job finding websites.

In chapter 4, this report describes in detail the architecture of the solution framework while providing all the specifications and details of its implementation.

In chapter 5, this report concludes the findings.

2 Problem Statement and Framework Overview

The COVID-19 pandemic revealed how important it is to ensure people's ability to find employment during emergencies, especially the less unprivileged. The pandemic poses a threat to young people's educational and employment prospects, in addition to causing the loss of life and wealth. The term "underprivileged" refers to disadvantaged students, the unemployed, low-paid, and socially excluded (young) people who make up a significant portion of the population and are underprivileged and underdeveloped from every perspective in society. In general, the lack of education that characterizes this class of people makes them, among others, socially disadvantaged. The goal is to enable underprivileged young people the opportunity to look for employment within the EU and develop their talents by getting advantage of the available tools developed in this project to better explain their skills and qualifications with particular, accepted semantics. On the other hand, EU businesses will receive assistance in describing professions and particular employment within a semantic context that has been established. In addition, organizations that provide on-the-job training must adjust their programs and learning outcomes to meet the needs of EU citizens and workers.

The goal of the project is to develop a semantically rich architecture/framework based on established standards to automate or semi-automate the annotation of job descriptions and UoLs while providing a goal-oriented agent-based system that will run without human intervention to persuade employers to accept less-skilled young men/women, whereas the second will have a personal assistant that will identify his/her needs and support life-long training and professional development.

In the following deliverables, we will present pilot applications as proof-of-concept tools where candidates can upload their skills to build their professional profiles and then enrich them with additional educational content sourced from the web of data. By connecting concepts from the ESCO taxonomy with abilities on candidate profiles and learning outcomes from open digital resources, this experimental application will showcase the expected feasibility and potential benefits by linking concepts from the ESCO taxonomy with skills on candidate profiles and learning outcomes derived from open digital resources. More specifically, we believe that such an application can significantly support formal, informal, and lifelong learners in obtaining the necessary skills to enhance their certifications.

3 Data: Job Vacancies

The solution framework for the EU Company deals with the Job Vacancies or JVs of each company. A job vacancy refers to an employment opportunity within an organization that is currently open and needs to be filled by a suitable candidate. When a company or employer identifies a need for additional personnel or a replacement for an existing employee, they create a job vacancy to attract qualified individuals who can fulfill the responsibilities and requirements of the position.

The process of announcing a job vacancy typically involves advertising the opening through various channels to reach a wide pool of potential candidates. This may include online job boards, company websites, social media platforms, recruitment agencies, industry-specific publications, and even word-of-mouth referrals.

A comprehensive job vacancy announcement usually includes the following details:

- Job Title: The specific position or role for which the company is seeking a candidate.
- Job Description: A detailed description of the responsibilities, tasks, and duties associated with the role. This section outlines the expectations and scope of the position, providing candidates with a clear understanding of what the job entails.
- Requirements: The skills, qualifications, experience, and knowledge necessary to perform the job successfully. This section may include educational requirements, certifications, technical skills, specific software proficiency, language proficiency, or any other essential qualifications.
- Desired Qualifications: Additional skills, experiences, or characteristics that would be advantageous for the role, but may not be strictly required.
- Application Process: Information on how interested candidates should apply, including instructions for submitting resumes, cover letters, portfolios, or any other required documents. It may also include details about interviews, tests, or assessments that will be part of the selection process.
- Deadline: The date by which applications must be submitted. This allows candidates to know the timeframe within which they need to apply.
- Contact Information: The name, email address, or phone number of the person or department to whom applicants should direct their inquiries or submit their applications.

Job vacancies serve as an opportunity for job seekers to find suitable employment and for employers to identify qualified individuals who can contribute to the success of their organization. By providing a detailed job vacancy announcement, employers can attract candidates who possess the necessary skills and qualifications and have a genuine interest in the position, streamlining the recruitment process and increasing the likelihood of finding the right fit for the job.

From all the sections mentioned above, the only bullets necessary for our framework's functionality are the requirements and the desired qualifications of the vacancy. The reason behind that is the structure of the ESCO vocabulary, which uses Skills and Occupations as the common vocabulary between jobs, courses and CVs, in order for the matches to happen between them.

Details on the acquisition of the JVs for our database can be found in deliverable 2.4.

4 Solution Framework

This section presents the architecture of the solution framework that was built in the framework of this project while discusses the implementation details of it.

4.1 Architecture

The implementation of the solution framework consists of several modules as presented in the list and the accompanying figure below. In the following list, a short description of each module is given. Information flows through the modules in the order they are listed below.

1. **Web Crawler**: This module is responsible for scraping websites containing jobs and extracting any information from them that are useful to our framework.
2. **JVs Database**: All JVs stored in the database has undergone annotator review prior to insertion in the database , enabling us to extract relevant rules.
3. **Text Extraction Module**: This module is responsible for extracting text from text files and then applying all necessary steps to the input, such as converting all text to lowercase, tokenizing, etc.
4. **Rules and Filters**: This module meticulously examines the JVs stored in the database to extract valuable rules and exceptions, which will be utilized as filters during the annotation process.
5. **Annotator module**: : In this module a variety of different processes take place as the annotator is the main component of the solution framework. The annotator's central task in this module is to extract skills and occupations from the candidate's JVs employing standard NLP methods and the ESCO terminology.
6. **Post-processing module**: In the final step of the implementation, the extracted skills and occupations undergo organization and refinement (including removing duplicates) before being presented to the user.

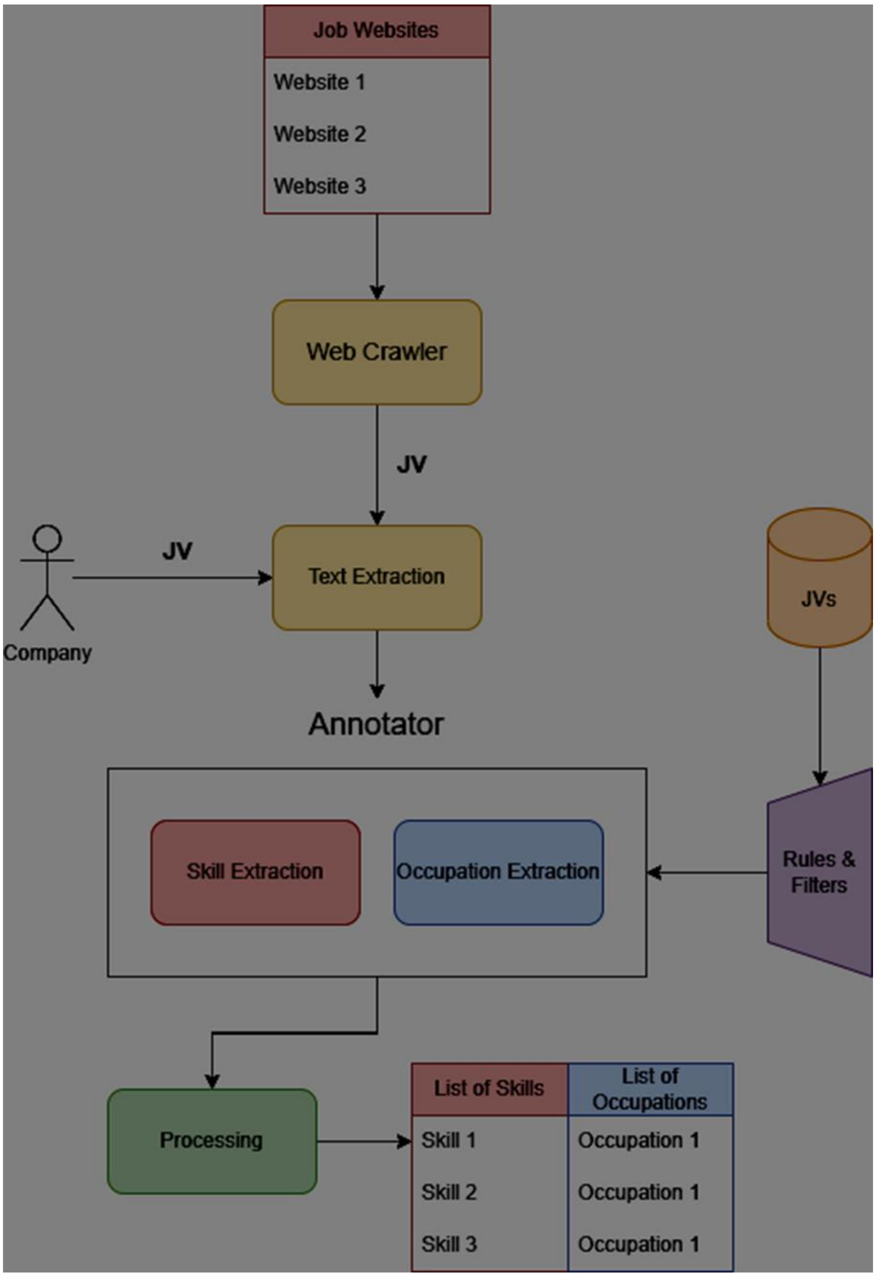

Figure 1: Architecture of Solution Framework.

4.2 Implementation

Before getting into the details of the main function of the framework, which is the processing of each JV and the extraction of the Skills and Occupations from it, we will describe in detail the function of the Web Crawler module. The module's role in the framework's organization is represented in Figure 1. The main aim of the Web Crawler module was to fill out the database with jobs so that:

1. We could be able to extract rules from them and use them as filters in the annotator and as a result improve its performance.

2. There are already available jobs in our system's database so that the candidate can search for jobs and jobs can be proposed to him/her by our system, regardless of the jobs the companies have inserted into the database.

Below are described the steps of the algorithm we used for scraping each of the websites. This algorithm follows a common structure for all of the websites. The first step is to find a link to the list of the job positions available in the website. Then the page that this link pointed to is scraped, and the tool tries to get a list of job pages that contained all the info necessary to us. After obtaining the content of each job position, we extracted and processed the data to obtain the following information:

- Title: This feature's value is the title of the job as scraped from its webpage.
- Provider: This feature's value is the name of the company that provides the specific job.
- Link: This feature's value is the URL of the job's webpage.
- Language: This feature's value is the language used in the job's description.
- Description: This feature's value is the job's description as found in its webpage.

For some of the websites scraped there were some exceptions added by hand in the algorithm. For example in the https://www.skywalker.gr/ website we needed to add some exceptions for some companies (e.g. Kotsovolos) because the pages of the job positions available in the website were different than all the other job pages. This type of occurrence was common in all websites scrapped so we added those kind exceptions in all of them.

All the scraped jobs were then organized into csv files depending on the website they were scraped from and the language their text was written in. Each listing in any of these files was a job with the values described above. Each of those files was then processed in order to clean the data in them and bring it in a uniform format that would later be given as input to the annotation algorithm. The steps of this processing method applied to each file, are the following:

1. Drop any listings that didn't contain any scraped description because this feature is necessary for the skill and occupation extraction module.
2. Clean the text in each field from any unnecessary characters and make it more uniformly formatted in order to make the extraction algorithm perform faster and improve its robustness.
3. Finally, if a file contained listings of multiple languages, we created separate files one for each language found in the in initial file.

The data contained in those files were finally inserted into our database. The process and methods used for inserting those data are going to be described in a later deliverable.

Below we will provide the details of the main module of the framework which is the annotator module. In the following paragraphs, we will present one-by-one the steps of the algorithm we built in order to extract ESCO skills and occupations from a given JV text.

The first step is to load the ESCO vocabulary into our system and, as mentioned earlier, keep only the attributes we need, namely preferred and non-preferred terms. Each language in our system has a different name. This means that for each language, the preferred term for a particular occupation will be different, even though the occupation behind it is the same.

The next step is to set up a list of skill and occupation exceptions for each language. We chose this because after doing a lot of experiments with the data, we found that some preferred and non-preferred terms, be they skills or occupations, are very common or common words in their respective languages. For example, the term "energy" is an ambiguous term that can be interpreted differently depending on the context, and it also appears frequently in the text. As a result, almost every JV is assigned a skill called "energy" without adding any useful information, which also makes it more difficult for our system's algorithms to work with this data. Therefore, we decided to create an exception list for each language to exclude such labels from the returned output of the algorithm.

After initializing the above necessary variables, we proceed to process the input text. For purposes of convenience, the processing steps are listed below in order.

1. Provide the text to be processed as input to the algorithm.
2. All texts are converted to lowercase, because all tags, be they skills or professions, are lowercase. Next, remove all non-alphanumeric characters from the string.
3. The transformed text is passed through a tokenizer in Python's spaCy library. Choose a different tokenizer for each language. Tokenization is the process of breaking down a large block of text into smaller pieces called tokens, such as words, phrases, symbols, or even subwords.
4. We filter out all tokens that fall into the exception list. We also filter those that contain numbers or other non-lexical symbols, except for some special characters (. # + * -), since ESCO labels are almost purely alphabetic, we need to reduce the number of tokens to speed up the search process.
5. Sequences of length 2 or 3 are created from the tokens produced in the earlier phases. Since most of ESCO labels only contain two or three words, this step is essential.
6. Following that, we begin looking for matches between the labels of the skills and the tokens or sequences. We add the corresponding skill to the output list for each match that is discovered.
7. The same procedure is repeated for the ESCO occupations.
8. Lastly, we eliminate any duplicate labels and deliver the output lists.

The libraries and their versions used in this algorithm’s implementation are listed in the following table.

Table 1: Libraries and their Versions used in the framework's implementation

Library	**Version**
Python	3.8
nltk	2.9.0
spaCy	3.3.1
pdfminer-six	2.1.3
pandas	1.4.3
selenium	4.10
beautifulsoup4	4.12.2
urllib	2.0.3

In the next two paragraphs we give some short descriptions of the main libraries used for scraping.

BeautifulSoup4 is a Python library used for web scraping and parsing HTML and XML documents. It provides a convenient way to extract data from web pages by traversing the HTML or XML structure and accessing specific elements based on their tags, attributes, or contents. BeautifulSoup4 is built on top of an HTML or XML parser, such as lxml, html5lib, or the built-in Python parser. It allows you to work with web data in a structured and flexible manner. You can use it to extract specific data points, navigate the document tree, modify elements, or generate new HTML/XML documents. Some common use cases of BeautifulSoup4 include extracting data from websites, scraping information for data analysis or research purposes, automating repetitive web tasks, and building web crawlers or spiders.

Selenium is a popular open-source framework for automating web browsers. It provides a way to interact with web pages, simulate user actions, and automate web testing or repetitive tasks. Selenium supports various programming languages, including Python, Java, C#, Ruby, and JavaScript, making it widely accessible for developers across different platforms.

Here are some key features and use cases of Selenium:

- Web Testing: Selenium is primarily used for web testing. It allows developers to write scripts that can simulate user interactions with web applications, such as clicking buttons, filling forms, navigating between pages, and verifying the expected behavior of the application. These scripts can be run across different browsers and platforms to ensure compatibility and functionality.
- Browser Automation: Selenium can automate repetitive tasks in web browsers, such as filling out web forms, scraping data from websites, taking screenshots, and downloading files. By

automating these tasks, developers can save time and effort, especially when dealing with large amounts of data or performing repetitive actions.

- Cross-browser Testing: Selenium supports multiple web browsers, including Chrome, Firefox, Safari, Internet Explorer, and Edge. This enables developers to test their web applications across different browsers and ensure consistent behavior and appearance.
- Web Scraping: Selenium can be used for web scraping, which involves extracting data from websites. It can navigate through web pages, interact with elements, and extract information using its API. Selenium is often combined with other libraries, such as BeautifulSoup, to extract specific data points from web pages.
- Web UI Development: Selenium can assist in the development of web user interfaces (UI). It allows developers to automate the UI testing process, ensuring that the UI elements and interactions work as intended.

5 Conclusion

This deliverable provided a detailed description of the development process for the solution framework for EU company. More specifically, we provided a short summary of the problem’s definition and the goal of the framework. We discussed details about all the necessary data for our framework to operate, which are JVs scraped from multiple job finding websites. Finally, we presented the architecture of the solution framework and provided all the specifications and details of its implementation.

References

[1] BRIDGING THE TRANSITION BETWEEN EDUCATION AND THE LABOUR MARKET, Report of the conference of the European Network of Education Councils, Prague, 20-21 October 2014.